Machine Learning with Graphs (MLG)

Homework 3 Report

吳岱桀 E14051350

1 INTRODUCTION

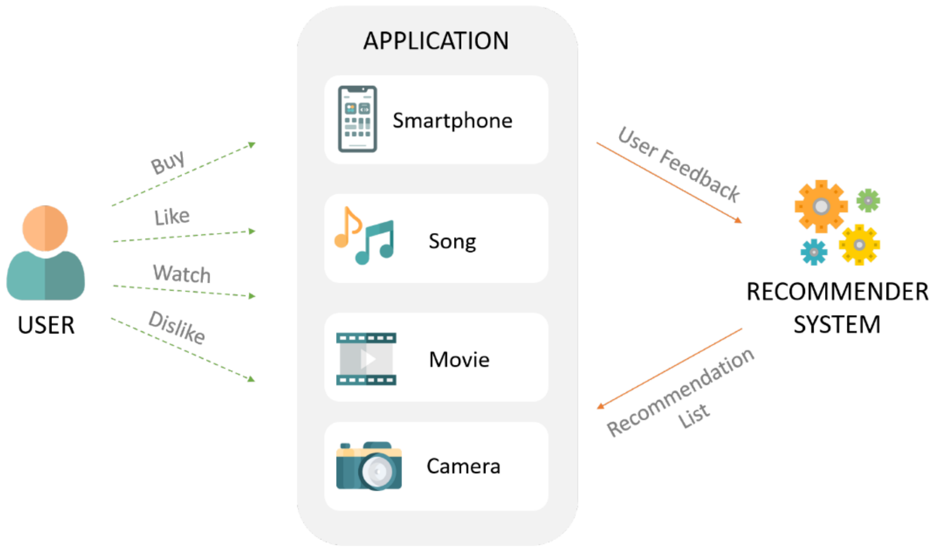
During the last few decades, with the rise of YouTube, Amazon, Netflix and many other such web services, recommender systems have taken more and more place in our lives. A recommender system is a type of data filtering approach using machine learning algorithms to recommend the most relevant items to a particular user or customer (items being movies to watch, text to read, products to buy or anything else depending on industries). It operates on the principle of finding patterns in consumer behavior data, which can be collected implicitly or explicitly.

For streaming service like Netflix, they use a recommender system to present viewers with movie and show suggestions. For e-commerce company like Amazon, on the other hand, they use a recommender system to present customers with product recommendations. While each uses one for slightly different purposes, both have the same goal: to drive sales, boost engagement and attention from users, and deliver more personalized customer experiences. Recommender systems are really critical in some industries as they can generate a huge amount of income when they are efficient or also be a way to stand out significantly from competitors

In this project, we will implement classic algorithm from the basic Collaborative Filtering to state-of-the-art deep learning approach on three datasets. Besides, with the help of Graph Neural Network (GNN), a combined method has proposed in this project. In the following sections, for each of method used, we will go through how they work and discuss their strengths and weaknesses.

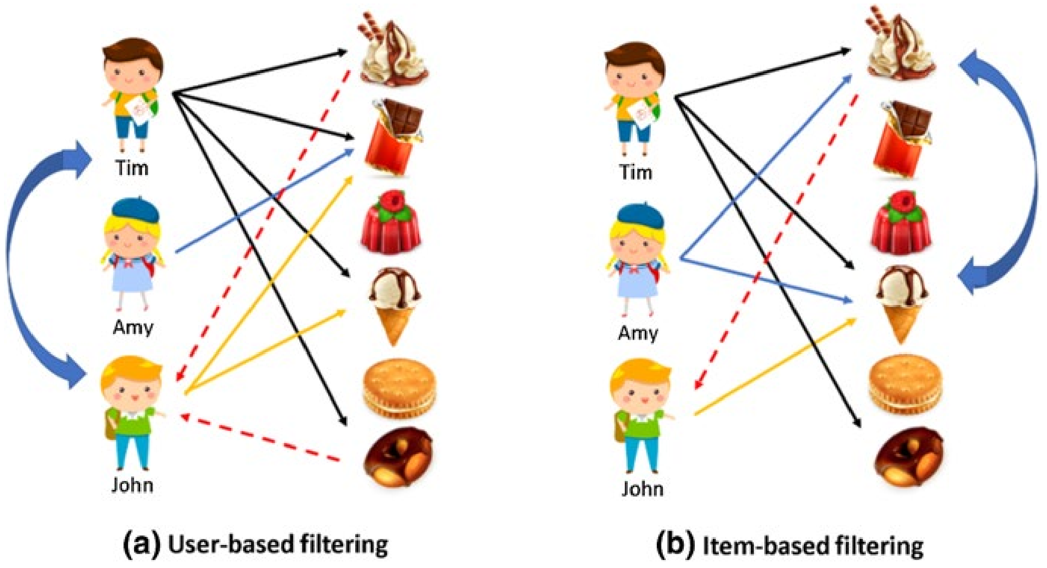
2 METHODOLOGY

Based on the user-item interaction history, user feature and item feature provided, the goal of a recommender system is to recommend the new item to users that they may be interested in. In this project we formulate the recommending task as a regression problem that we would have to predict the rating of item given by user. A simple illustration is shown as follows:



2.1 Collaborative Filtering (CF)

In this part, we'll introduce the method of collaborative filtering, which divided in two major approach to calculate the similarity among users or items as user-based collaborative filtering and item-based collaborative filtering.



There are two common similarity metrics used in collaborative filtering. That is, Pearson correlation and cosine similarity. The formula is shown as follows:

Cosine Similarity:

$$\text{sim}(u, v) = \cos(r_u, r_v) = \frac{r_u \cdot r_v}{\|r_u\| \|r_v\|}$$

Pearson Correlation Coefficient

$$sim(u,v) = \frac{\sum_{i \in S_{uv}} (r_{ui} - \overline{r_u})(r_{vi} - \overline{r_v})}{\sqrt{\sum_{i \in S_{uv}} (r_{ui} - \overline{r_u})^2} \sqrt{\sum_{i \in S_{uv}} (r_{vi} - \overline{r_v})^2}}$$

2.1.1 User-based Collaborative Filtering

The goal of user-based collaborative filtering is to find similar users of a certain user. Item's rating is predicted based on how similar items have been rated by that user. The ratings are predicted using the user's own ratings on closely related items.

The rating of a certain item that given by the user can be computed as follows:

$$r_{ui} = \frac{\sum_{v \in N} s_{uv} \cdot r_{vi}}{\sum_{v \in N} s_{uv}}$$

The advantage of user-based collaborative filtering is that it is easy to implement and context independent. Performance of user-based method would keep improving as neighborhood K sizes are increased. However, there exist sparsity and cold-start issue because the percentage of people who rate items tend to be really low and new users will have nearly no information of rating to be compared with other users. Besides, scalability might become a problem as the more users there are in the system, the greater the cost of finding the nearest K neighbors will be.

2.1.2 Item-based Collaborative Filtering

The goal of item-based collaborative filtering is to find similar items of a certain item. Item's recommendation rating for a user is calculated depending on that items' ratings by other similar users. Neighborhoods are defined by similarities among items.

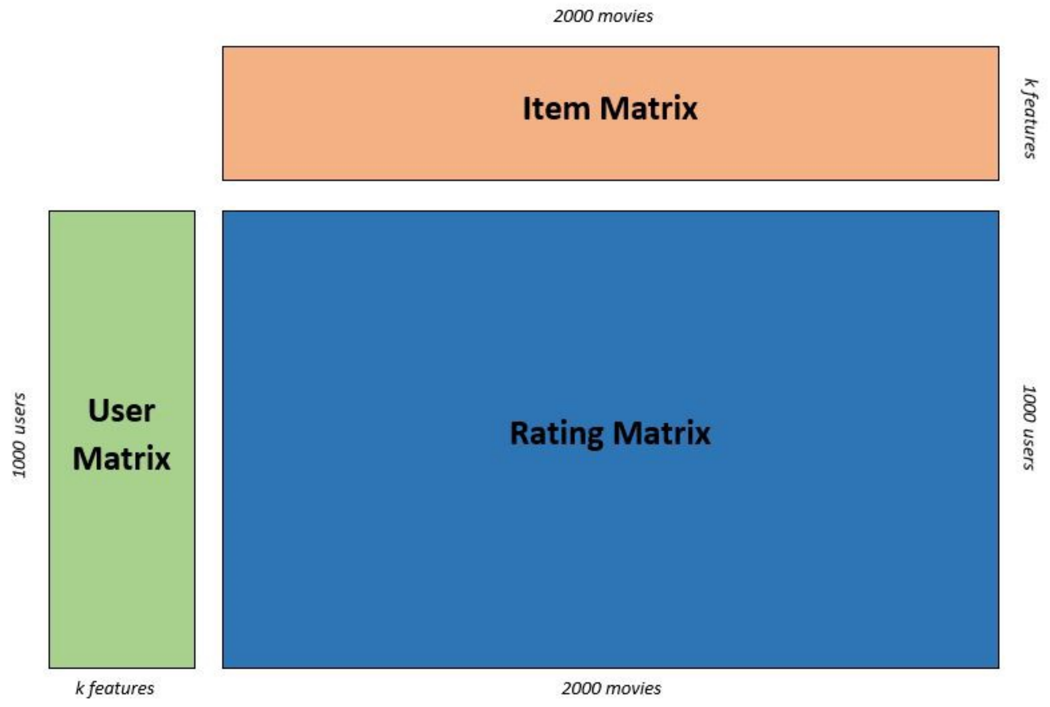
The rating of a certain item that given by the user can be computed as follows:

$$r_{ui} = \frac{\sum_{j \in N(i;u)} s_{ij} \cdot r_{uj}}{\sum_{j \in N(i;u)} s_{ij}}$$

Where S_{ij} similarity of items i and j, r_{ui} is rating of user u on item j and $N(i; u)$ is set of items rated by u similar to i.

Since the ratings are predicted using the ratings of user herself, the predicted ratings tend to be much more consistent with the other ratings, which the prediction accuracy of item-based collaborative filtering is usually better than user-based collaborative filtering. On the other hand, item-based methods might sometimes recommend obvious items, or items which are not novel from previous user experiences.

2.2 Matrix Factorization (MF) and Factorization Machine (FM)



Factorization is the method of expressing something big as a product of smaller factors. Matrix Factorization finds two rectangular matrices with smaller dimensions to represent a big rating matrix (RM) while those small matrices can keep the properties of the rating matrix. One matrix can be seen as the user matrix (UM) where rows represent users and columns are k latent factors. The other matrix is the item matrix (IM)where rows are k latent factors and columns represent items. Here k < number of items and k < number of users. The latent matrices treated as features and features are characteristics of an item or user. To predict the whole rating matrix by user and item matrix, it can be calculated as follows:

$$\hat{Y} = UV^T$$

The process to refine the latent user and item matrix can be formulated using gradient decent to minimize the mean squared error (MSE) over the matrices. The loss function is defined as follows:

$$E(U,V) = \frac{1}{N} \sum_{(i,j):r_{ij}=1} (y_{ij} - u_i \cdot v_j)^2$$

The gradient descent equations are:

$$u_{ik} \leftarrow u_{ik} + \frac{2\eta}{N} \sum_{j:r_{ij}=1} (y_{ij} - u_i \cdot v_j) v_{jk}$$

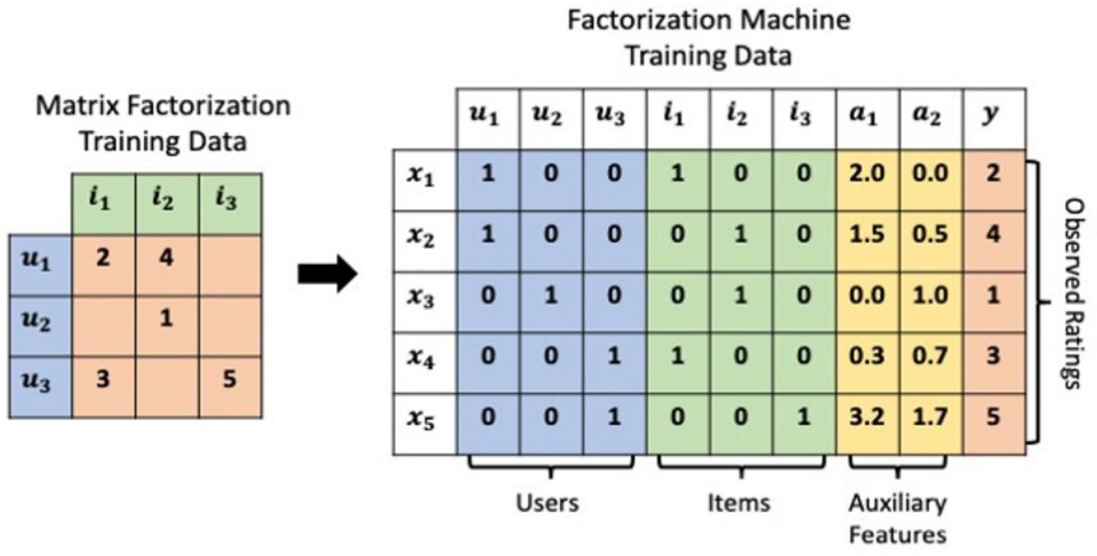
$$v_{jk} \leftarrow v_{jk} + \frac{2\eta}{N} \sum_{i:r_{ij}=1} (y_{ij} - u_i \cdot v_j) u_{ik}$$

The regularization term can be added to avoid overfitting in training process. The regularized loss function is:

$$\frac{1}{N} \sum_{(i,j):r_{ij}=1} (y_{ij} - u_i v_j)^2 + \lambda(\sum_{i=1}^{n_u} \sum_{k=1}^{K} u_{ik}^2 + \sum_{i=1}^{n_m} \sum_{k=1}^{K} v_{ik}^2)$$

Where $N = \sum_{ij} r_{ij}$

Unlike the classic MF model discussed above which inputs a user-item interaction matrix, Factorization Machine models represent user-item interactions as tuples of real-valued feature vectors and numeric target variables. The limitation of CF and MF is that those methods can only make use of the user-item interaction matrix. However, other than user-item interaction matrix merely containing interaction history, the auxiliary features is able to be incorporated in FM. For example, user or item features and contextual features relevant to the interaction itself may be adopted with the user-item interaction matrix together.



	i_1	i_2	i_3
u_1	2	4	
u_2		1	
u_3	3		5

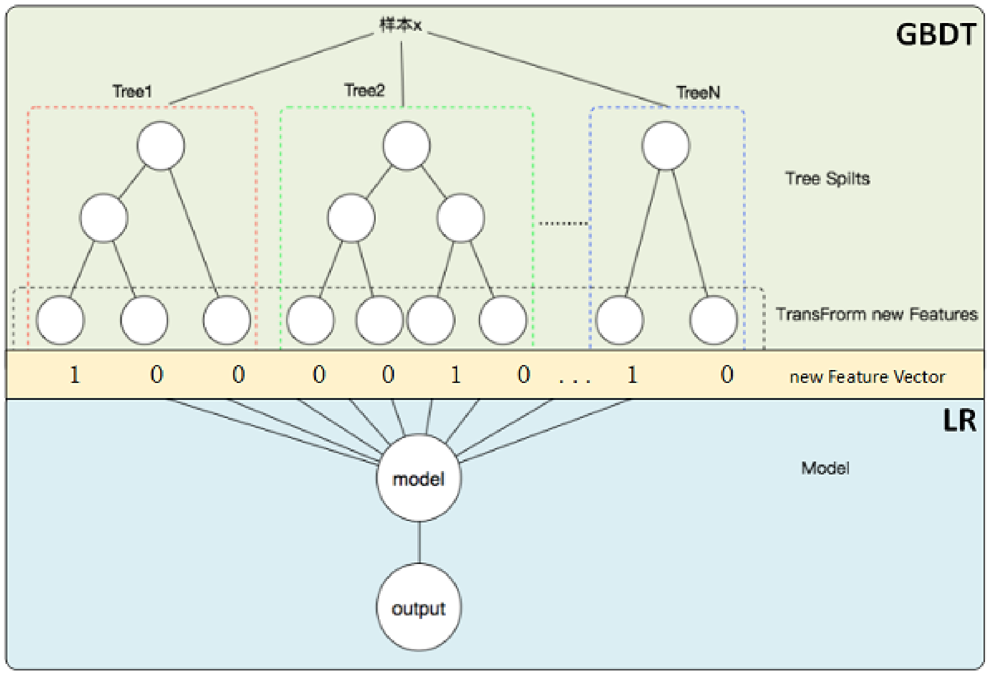
	u_1	u_2	u_3	i_1	i_2	i_3	a_1	a_2	y
x_1	1	0	0	1	0	0	2.0	0.0	2
x_2	1	0	0	0	1	0	1.5	0.5	4
x_3	0	1	0	0	1	0	0.0	1.0	1
x_4	0	0	1	1	0	0	0.3	0.7	3
x_5	0	0	1	0	0	1	3.2	1.7	5

The FM model equation is comprised of n-way interactions between features. A second-order model (by far the most common) includes weights for each base feature as well as interaction terms for each pairwise feature combination. Unlike polynomial linear models which estimate each interaction term separately, FMs instead use factorized interaction parameters: feature interaction weights are represented as the inner product of the two features' latent factor space embeddings. This greatly decreases the number of parameters to estimate while at the same time facilitating more accurate estimation by breaking the strict independence criteria between interaction terms. The FM model can be formulated as follows:

$$f(x) = w_0 + \sum_{p=1}^{P} w_p x_p + \sum_{p=1}^{P-1} \sum_{q=p+1}^{P} \langle v_p, v_q \rangle x_p x_q$$

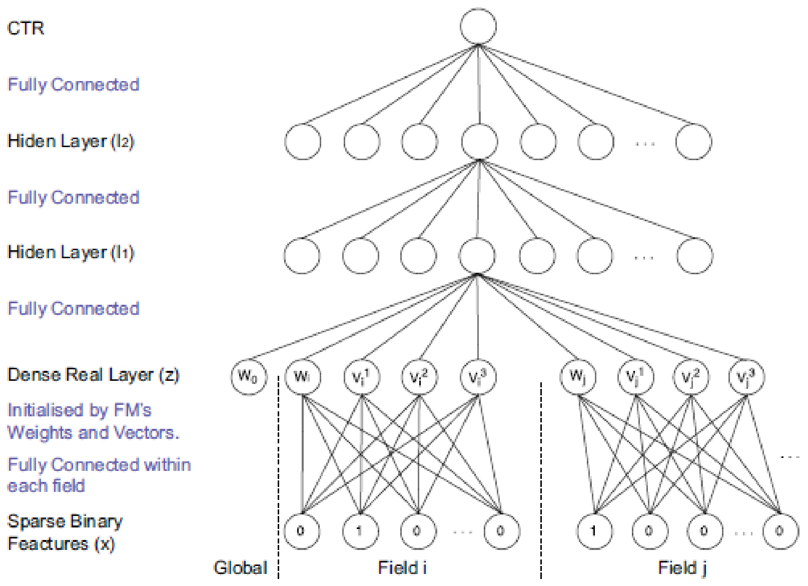
2.3 GBDT + LR

In this part, we'll introduce the method of applying GBDT (Gradient Boosting Decision Tree) with regression. Regression method is easy to implement in most of the use case. However, with the feature property in the domain of recommender system tends to be sparse (categorical features have too many classes in a single feature), It would be hard to directly employ regression method for recommender system. Besides, the cross features cannot be captured in simple regression.



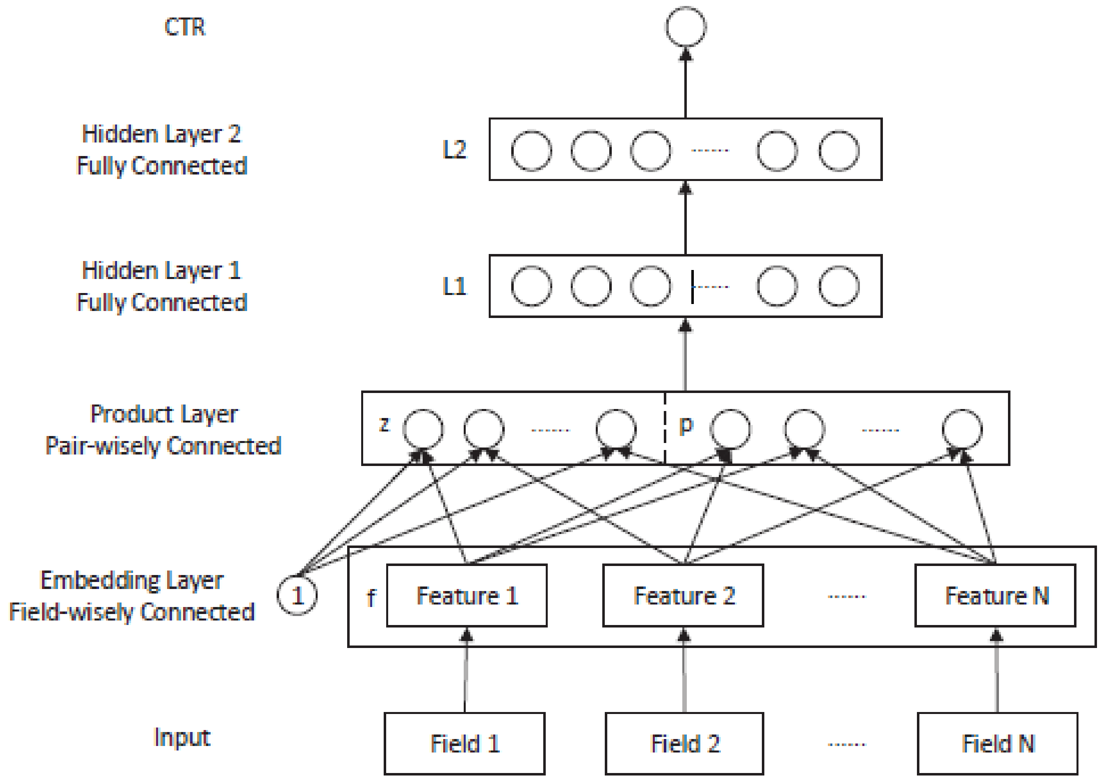
GBDT is introduced to solve the problem mentioned above. The pre-trained GBDT is used to obtain cross features. Each leaf in a tree generates a cross feature and each cross feature represent as one-hot encoding. The obtained multi-hot encoding feature vector then is treated as the input feature of linear regression to map rating score or logistic regression for the application of prediction of Click-Through Rate (CTR).

2.4 FM-supported Neural Networks (FNN)



The input features in recommender system are usually of multi-field and are mostly discrete and categorical that are different from continuous raw features that we tend to use in other domains. In previous research, some works mainly focus on linear models that may lose the ability of exploring feature interactions. High-order combination features can be built up manually, but it requires heavy computation in the large feature space. It's proposed in FNN to use embedding dense feature to automatically learn effective patterns from categorical feature interactions

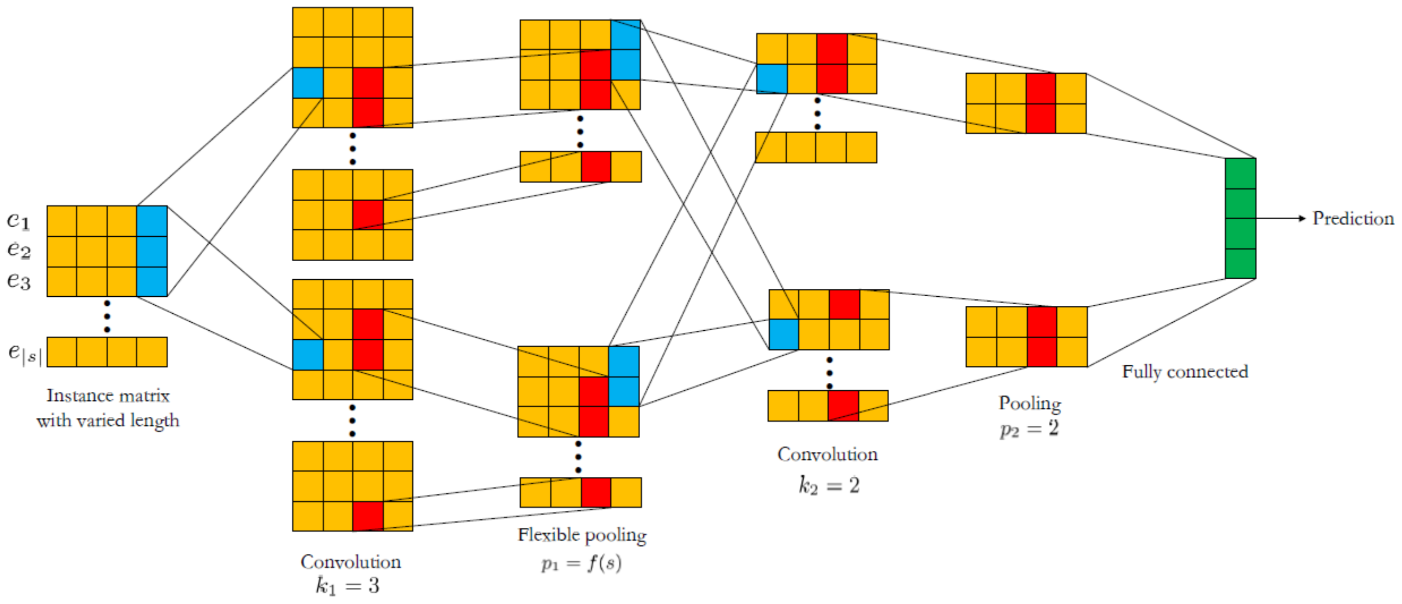
2.5 Inner Product Neural Network (IPNN) and Outer Product Neural Network (IPNN)



Similar to FNN trying to deal with interaction among features, it's proposed in Product-based Neural Networks (PNN) to use an embedding layer to learn a distributed representation of the categorical data, a product layer to capture interactive patterns between different categories, and fully connected layers to obtain high-order feature interactions. The main contributions of PNN are conquering the problem that low-order feature combinations may not be powerful enough and simple MLP is not able to take high-dimensional features.

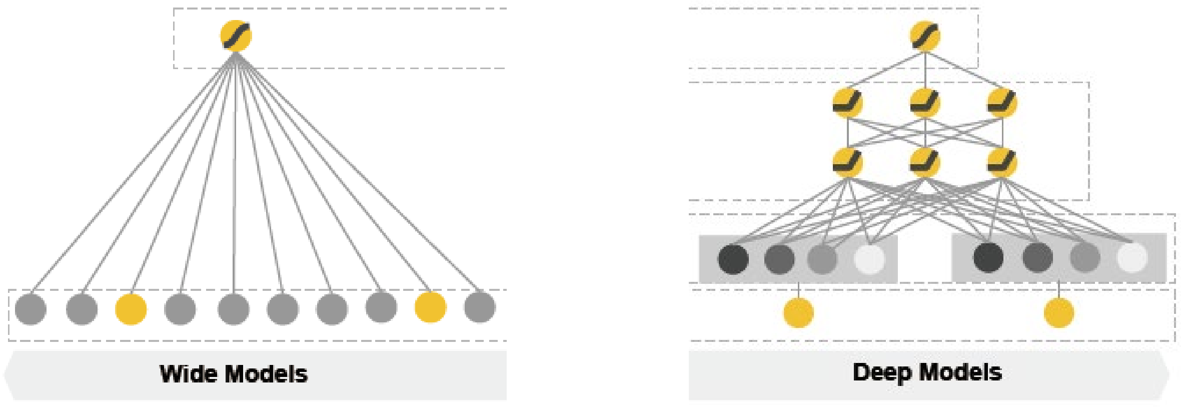
The difference of IPNN and OPNN is at product layer, which employs inner-product or outer-product operation on features. Inner-product operation would output a vector while outer-product operation would output a matrix, so it is necessary to further flatten the matrix into 1-d vector before it is passing through fully connected layers.

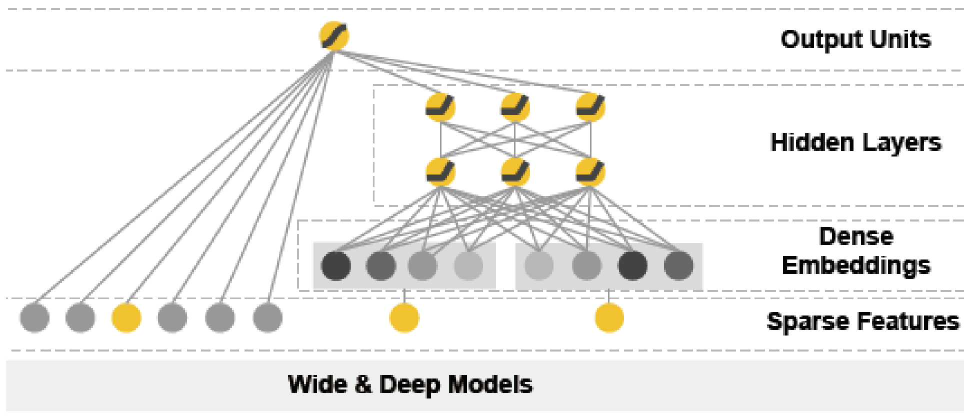
2.6 Convolutional Click Prediction Model (CCPM)



Convolutional Click Prediction Model (CCPM) is proposed to extract local-global key features from an input instance with varied elements based on convolution neural network. It applies convolution operation on embedded feature vectors and generate max-pooled feature maps, which is then passed to fully connected layer.

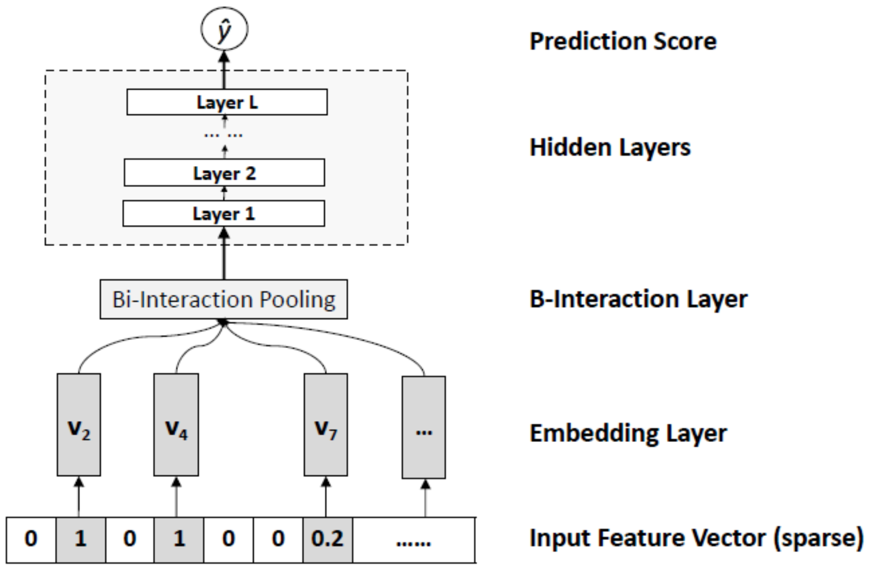
2.7 Wide & Deep





Wide & Deep is proposed to learn jointly trained wide linear models and deep neural networks to combine the benefits of memorization and generalization for recommender systems. Deep neural networks can generalize better to unseen feature combinations through low-dimensional dense embeddings learned for the sparse features with less feature engineering, while it may cause over-generalizing problem when user-item interactions are sparse. As for the cross-product feature transformations, it's effective and interpretable to be expressed as interaction feature and it has the property of memorization.

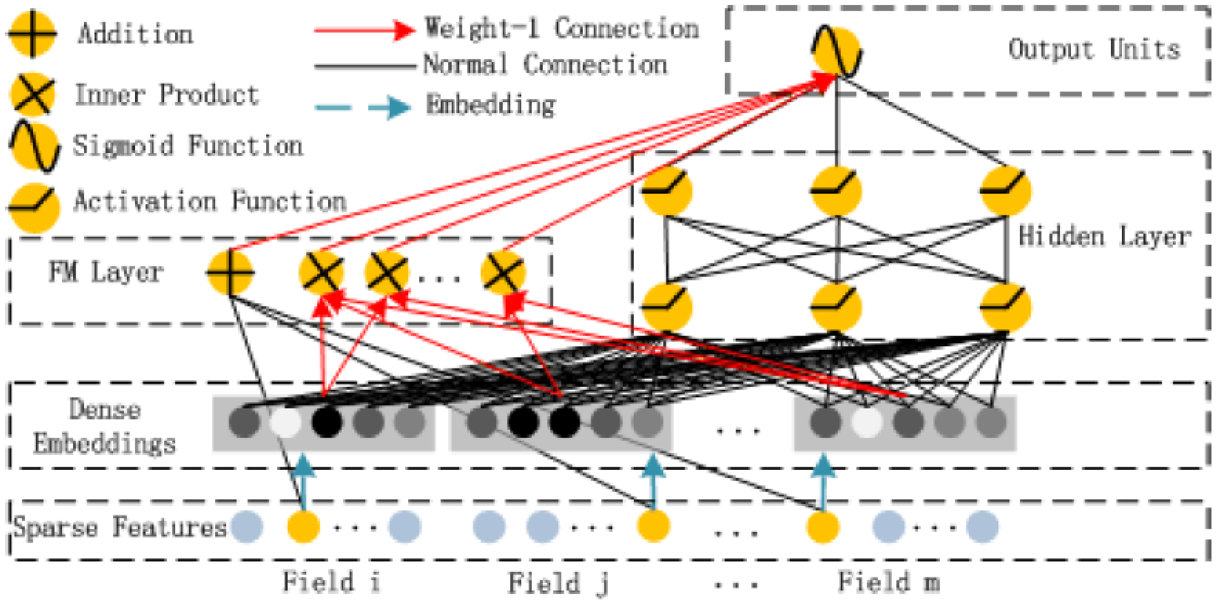
2.8 Neural Factorization Machine (NFM)



Neural Factorization Machine (NFM) is proposed for prediction under sparse settings. NFM combines the linearity of FM in modelling second-order feature interactions and the non-linearity of neural network in modelling higher-order feature interactions. NFM is more expressive than FM because FM can be seen as a special case of NFM without hidden layers. The categorical features would be first passed through a embedding layer and then a Bi-Interaction layer would be adopted to take embedded feature as input.

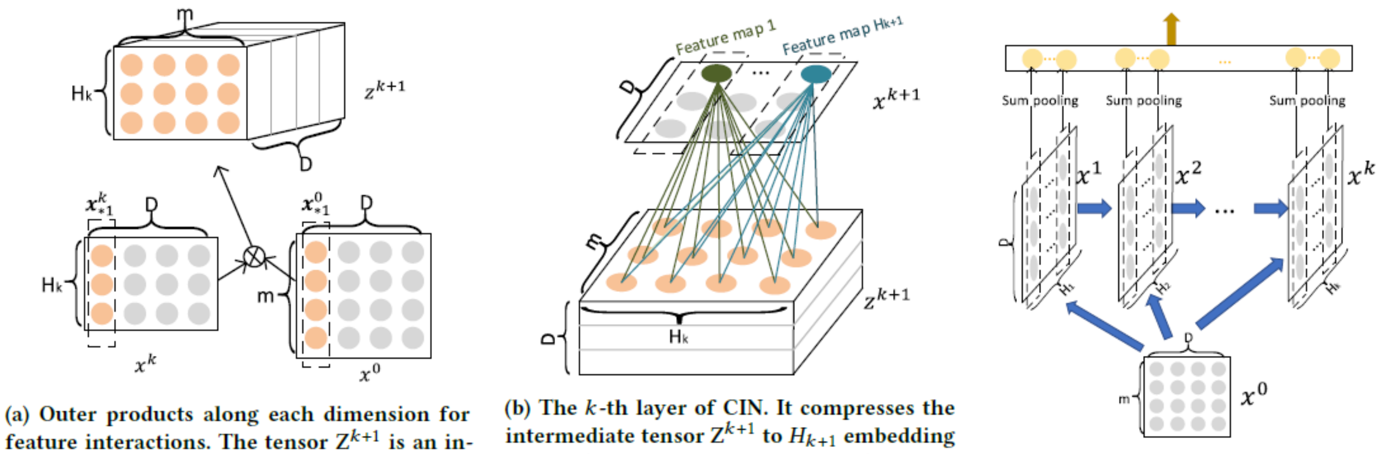
$$f_{BI}(\mathcal{V}_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i \mathbf{v}_i \odot x_j \mathbf{v}_j,$$

2.9 DeepFM



DeepFM is proposed to emphasize both low- and high-order feature interactions. It combines the power of FM for recommendation and deep learning for feature learning. In the FM part, it takes inner-product operation to obtain reach the same ability of feature interaction in FM.

2.10 eXtreme Deep Factorization Machine (xDeepFM)



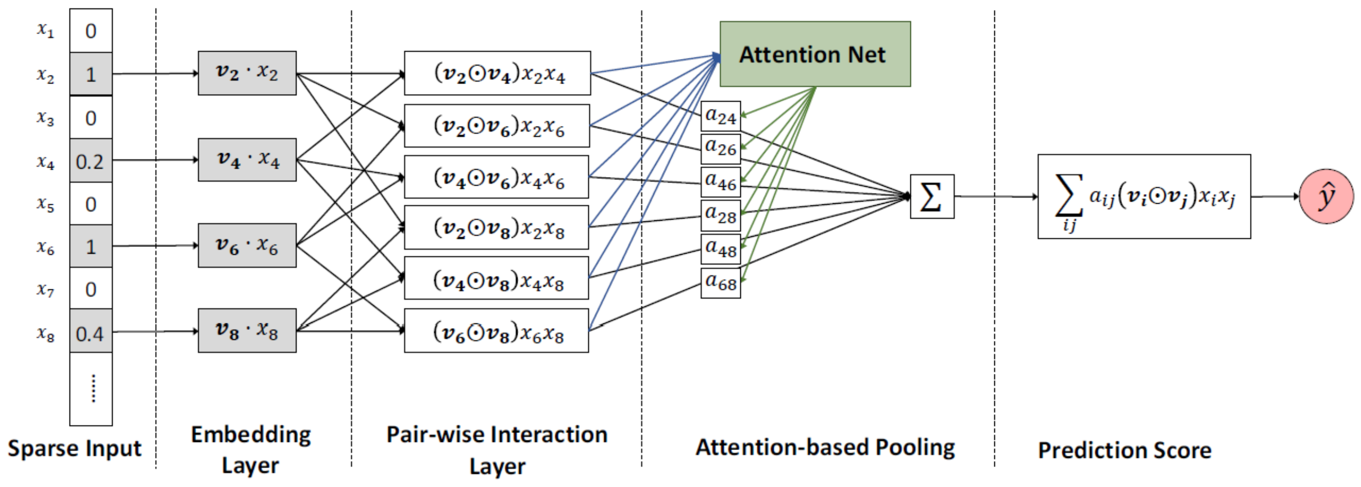
(a) Outer products along each dimension for feature interactions. The tensor Z^{k+1} is an intermediate result for further learning.

(b) The k-th layer of CIN. It compresses the intermediate tensor Z^{k+1} to H_{k+1} embedding vectors (aslo known as ***feature maps***).

(c) An overview of the CIN architecture.

As an improved version of DeepFM, a novel Compressed Interaction Network (CIN) is proposed in eXtreme Deep Factorization Machine (xDeepFM) to generate feature interactions at the vector-wise level explicitly. It's mentioned that CIN share some functionalities with CNN and RNN. xDeepFM is consisted of a CIN and a classical DNN, which is able to learn low- and high-order feature interactions implicitly.

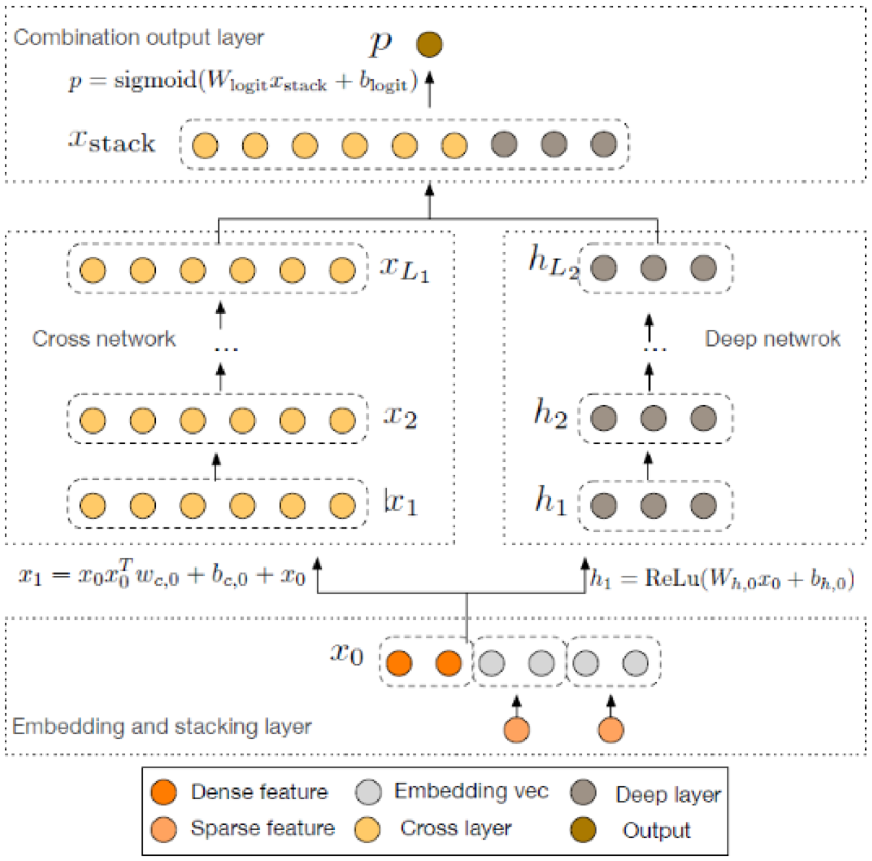
2.11 Attentional Factorization Machines (AFM)



Although FM is proved to be effective in previous works, the process to consider all feature interactions to be equally useful with the same weight may be problematic. The interactions obtained from useless features is probably degrading the performance because of the noises introduced by the interaction operation. As a result, it's crucial to distinguish the importance of different feature interactions by given them different weight. In Attentional Factorization Machine (AFM), it learns the importance of each feature interaction from data by adopting the concept of attention mechanism, which Attention-based Pooling Layer is incorporated in the model as follows:

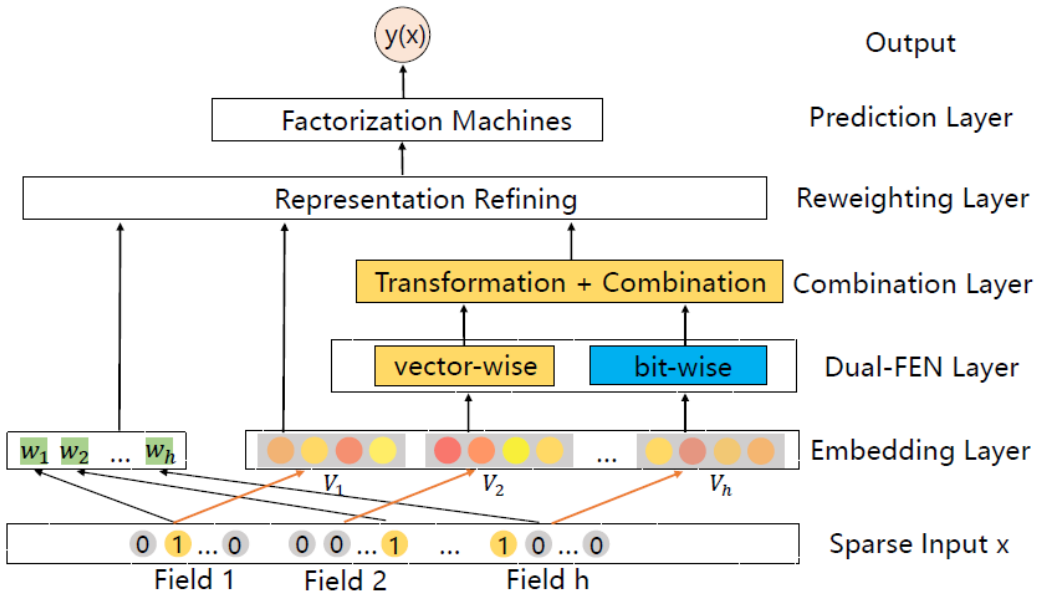
$$f_{Att}(f_{PI}(\mathcal{E})) = \sum_{(i,j)\in\mathcal{R}_x} a_{ij}(\mathbf{v}_i \odot \mathbf{v}_j)x_i x_j$$

2.12 Deep & Cross Network (DCN)


Combination output layer
p
$p = \mathrm{sigmoid}(W_{\mathrm{logit}} x_{\mathrm{stack}} + b_{\mathrm{logit}})$
x_{stack}
x_{L_1}
Cross network
x_2
x_1
h_{L_2}
Deep netwrok
h_2
h_1
$x_1 = x_0 x_0^T w_{c,0} + b_{c,0} + x_0$
$h_1 = \mathrm{ReLu}(W_{h,0} x_0 + b_{h,0})$
x_0
Embedding and stacking layer
Dense feature
Sparse feature
Embedding vec
Cross layer
Deep layer
Output

Although DNNs are able to automatically learn feature interactions, it's sometimes not efficient to learn all types of cross features and it's done implicitly. In Deep & Cross Network (DCN), a novel cross network is proposed to learn certain bounded-degree feature interactions efficiently that still keeps the benefits of a DNN model. To be specific, DCN explicitly applies feature crossing at each layer automatically without manual feature engineering.

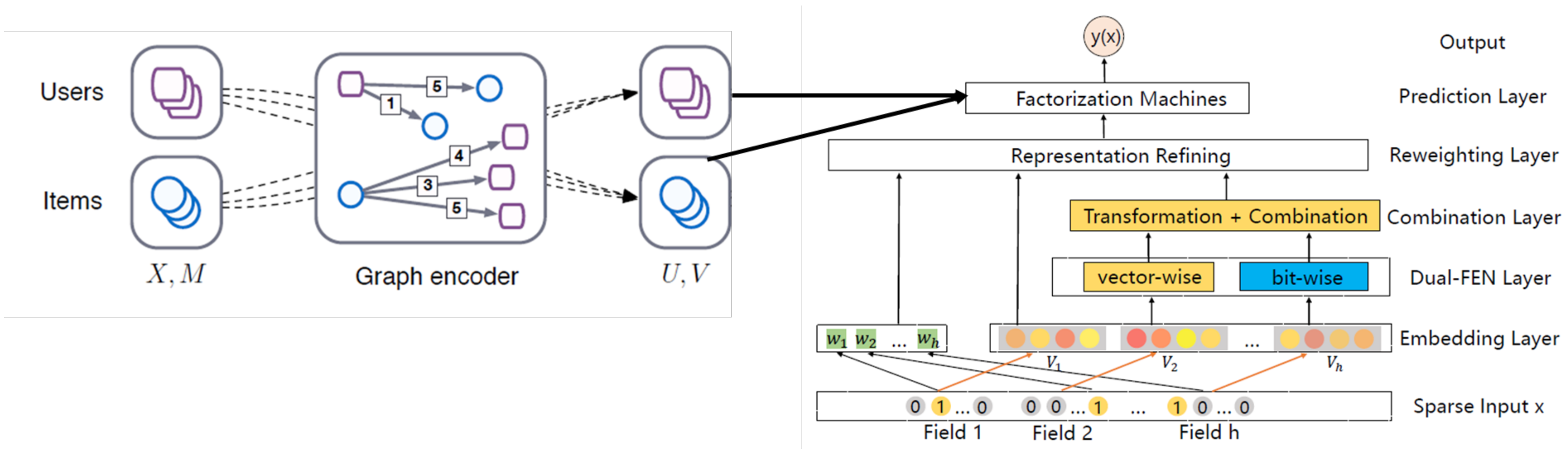
2.13 Dual Input-aware Factorization Machines (DIFMs)



Standard FMs produce only a single fixed representation for each feature across different input instances that may limit model's expressive and predictive power. Dual Input-aware Factorization Machines (DIFMs) is proposed to adaptively reweight the original feature representations at the bit-wise and vector-wise levels simultaneously, which makes it possible to learn a more flexible and informative representations of a given feature according to different input instances. Furthermore, Multi-Head Self-Attention, Residual Networks and DNNs are integrated into a unified end-to-end model to have better predictive power.

2.14 Proposed Method

In this part, we'll introduce a refine method to integrate a GNN based method Graph Convolutional Matrix Completion (GCMC) with DIFMs.



The core idea of GCMC is to encode the original user-item rating matrix into a user and item embedding by formulating rating matrix as a bipartite graph. To obtain the predicted rating in GCMC, a Bilinear decoder followed by the application of a softmax function is shown as follows:

$$p(\check{M}_{ij} = r) = \frac{e^{u_i^T Q_r v_j}}{\sum_{s\in R} e^{u_i^T Q_s v_j}}$$

Consider the process of predicting in GCMC can be treated as another form of inn-product operation on the user and item embedding, we propose to incorporate the obtained user and item embedding in GCMC in Prediction Layer of DIFMs, which user and item embedding can be considered as features containing more expressive topology information while the FM in Prediction Layer allows it to combine user and item embedding with other features obtained in previous layer together.

3 EXPERIMENT & ANALYSIS

To compare the performance of different approach, 5-fold cross evaluation is taking which randomly selects 20% user-item interactions as test data and the remaining data is split into training (70%) and validation (10%) sets. The performance is using average scores of RMSE, Recall, NDCG over 5 testing sets.

	MovieLens			**Yelp**			**Douban Book**		
	RMSE	**Recall @ 10**	**NDCG @ 10**	**RMSE**	**Recall @ 10**	**NDCG @ 10**	**RMSE**	**Recall @ 10**	**NDCG @ 10**
UCF-s	0.9586	0.7390	0.9536	1.1386	0.6723	0.9681	0.7169	0.9086	0.9789
UCF-p	0.9543	0.7666	0.9540	1.1505	0.6674	0.9641	0.7349	0.9035	0.9768
ICF-s	0.9476	0.7884	0.9549	1.1801	0.6841	0.9696	0.7102	0.9166	0.9795
ICF-p	0.9475	0.7956	0.9559	1.1912	***0.6927***	0.9700	0.7308	0.9063	0.9777
MF	0.9344	0.7722	0.9571	1.7360	0.4326	0.9707	0.7091	0.9036	0.9801
FM	0.9327	0.7654	0.9563	1.1460	0.6865	0.9684	0.6999	0.9064	0.9782
XGB+LR	0.9346	0.7768	0.9547	1.1621	0.6693	0.9711	0.6957	0.9089	0.9811
LightGBM+LR	0.9319	0.7732	0.9539	1.1591	0.6627	0.9684	0.6943	0.9113	0.9818
IPNN	0.9363	0.7778	0.9553	1.0589	0.6358	***0.9727***	0.6852	0.9187	0.9837
OPNN	0.9318	0.7964	0.9540	1.0646	0.6229	***0.9727***	0.6819	0.9164	0.9852
CCPM	0.9309	0.7901	0.9556	1.0923	0.6665	0.9714	0.6807	0.9108	0.9751
WD	0.9441	0.7784	0.9540	1.0588	0.6489	0.9726	0.6774	0.9205	0.9784
NFM	***0.9236***	0.8014	***0.9574***	1.0576	0.6452	0.9725	0.6735	0.9256	0.9849
DeepFM	0.9418	0.7898	0.9556	1.0887	0.6576	0.9713	0.6914	0.9269	0.9799
AFM	0.9330	0.7768	0.9551	***1.0571***	0.6879	0.9722	0.6741	0.9271	***0.9897***
xDeepFM	0.9475	0.7781	0.9537	1.0655	0.6552	0.9724	0.6879	0.9133	0.9825
DCN	0.9355	***0.8045***	0.9552	1.0917	0.6531	0.9718	0.6852	0.9217	0.9846
DIFM	0.9458	0.7619	0.9533	1.0742	0.6423	0.9722	***0.6728***	***0.9294***	0.9783
GCMC + DIFM	0.9259	0.7993	0.9659	1.0593	0.6861	0.9719	0.6716	0.9314	0.9856

The bold score with underline is the best one of that score. For example, 0.9236 is the best RMSE score in dataset of MovieLens. On the contrary, the red outer frame represents the approach with best performance which take all three metrices in consideration.

As we can see in the result, it's not necessary that an approach with highest score in a certain metric would also performs well in other metrices. Besides, the capability of models is not consistent which one model performs the best in Yelp dataset may not be the best method in other datasets. In general, NN models perform better than non-NN models. We analyze that there are two reasons that may result in better performance of NN models, which it captures interaction feature effectively and it's able to make use of all the auxiliary feature with the user-item interaction that CF and MF can't. Considering the time for training, we make all approach of NN models adopt same setting such as dimension of embedding layer and dropout rate, which it should be fine-tuned respectively to achieve the best performance in each of model.

As for the proposed method GCMC + DIFM, although it’s not the best method for all three datasets, it outperforms using DIFM solely. It shows that by incorporating the user and item embedding with expressive topology information, the combined method of GCMC and DIFM successfully utilize the advantage of GNN and the integrated framework in DIFM

4 CONCLUSION

In this project, we have tried different methods ranging form collaborative filtering, matrix factorization to recent proposed NN-based methods to deal with recommending task. We have again confirmed that Neural Network improves the ability of generalization and memorization from the reproduction of NN models mentioned in this project. Besides, we propose an ensemble method to employ Graph Neural Network with a new model Dual Input-aware Factorization Machines together, and it indicate the advantage to exploit the information form bipartite graph formulated in GCMC

5 REFERENCE

[1] User-based vs Item-based Collaborative Filtering | by Mustafa Katipoğlu | Recommendation Systems | Medium

[2] Recommender Systems — User-Based and Item-Based Collaborative Filtering | by Carlos Pinela | Medium

[3] Recommender Systems: Matrix Factorization from scratch | by Aakanksha NS | Towards Data Science

[4] Kaggle滑水 - CTR預估（GBDT-LR） - 台部落 (twblogs.net)